Prima Designdokument

Titel: DoomClone

Name: Florian Pfützenreuter

Matrikelnr.: 257119

1. Spielablauf

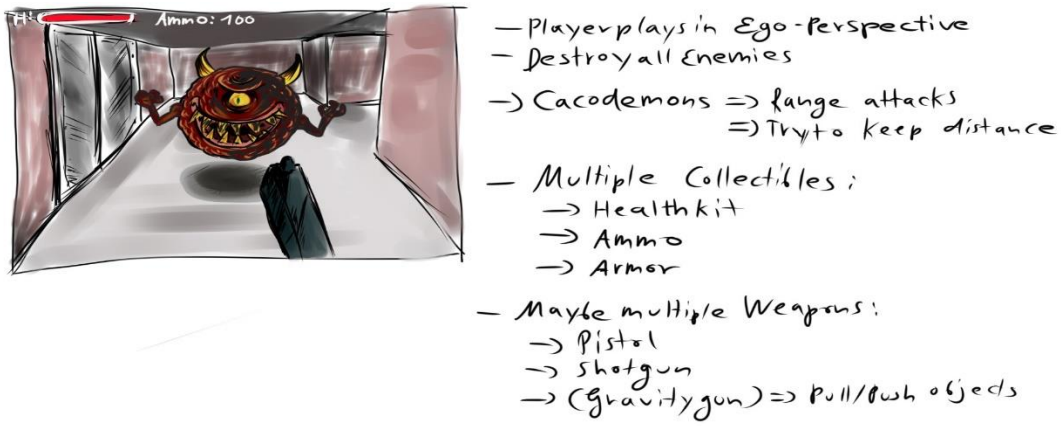

Abbildung 1: Anfängliche Konzeptionsgrafik

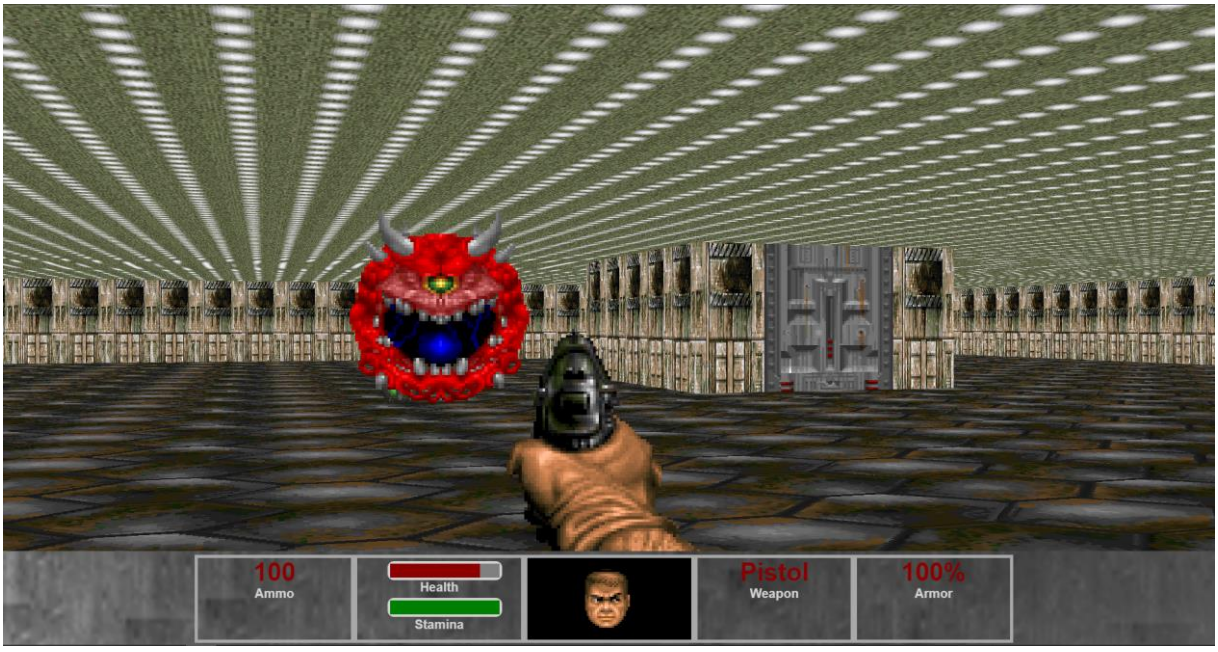

Abbildung 2: Fertiger Spielprototyp

Bei dem Spiel handelt es sich um einen Klon des Spiels Doom. Der Spieler befindet sich in der Egoperspektive. Ziel des Spiels ist alle Gegner zu töten, ohne selbst zu sterben. Hierzu bewegt sich der Spieler standardmäßig mit den Pfeiltasten, schießt mit der linken Steuerungstaste, sprintet mit der linken Shift-Taste und interagiert mit der Leertaste mit der Tür. Das Spiel kann mit der Escapetaste pausiert werden. Der Spieler kann 3 Items

aufnehmen: ein HealthKit, ein AmmoKit und ein ArmorKit. Gegner können den Spieler verfolgen, ihn beschießen und vor ihm fliehen. Die Tastenbelegung kann im Hauptmenü gewechselt werden.

2. Nutzerinteraktion

Der Nutzer kann mit dem Spiel interagieren, indem er zum einen die Tastenbefehle des Spiels selbst wählen kann auf der Menüseite „Controls“. Zum anderen kann durch diese Tastenbefehle die Spielfigur gesteuert, Gegner beschossen, Items aufgesammelt und die Tür geöffnet/geschlossen werden. Mit der Escapetaste kann das Spiel pausiert werden. Die Interaktionsmedien auf den Menüseiten sind Maus und Tastatur. Das Spiel selbst unterstützt nur Tastaturbefehle.

3. Objektinteraktion

Mithilfe von Objektkollisionen interagieren eine Vielzahl von Objekten in dem Spiel. So sendet der Spieler pro Bewegung/Rotation ein Event an die Wände bzw. die Tür. Sollte die Position des Spielers innerhalb der Kollisionssphäre liegen, wird die Position um die letzte Bewegungseinheit negativ in Z-Richtung verschoben. Projektile senden pro Bewegung auch eine Nachricht an alle Gegner/Spieler, Wände und die Tür. Sollte es zu einer Kollision kommen mit einer Wand oder der Tür, so wird die Explosionsanimation abgespielt und nach Ablauf dieser, das Projektil gelöscht. Sollte ein Gegner oder ein Spieler getroffen werden, nimmt der Gegner/Spieler schaden. Auch hier wird danach die Explosionsanimation abgespielt und danach das Projektil gelöscht. Sollte es zu keiner Kollision kommen bevor die definierte Range erreicht wurde, explodiert das Projektil und wird gelöscht. Gegner können auch mit Wänden und Türen kollidieren. Hierzu wird bei jedem Schritt auch eine Nachricht an alle Wände und die Tür versendet. Bei einer Kollision wird der Gegner um 90 Grad auf der Y-Achse rotiert und um die letzte Bewegungseinheit negativ auf der Z-Achse verschoben. Die Kollisionsabfrage bei der Tür erfolgt nur jeweils, wenn die Tür geschlossen ist.

4. Objektanzahl variabel

Die Projektile der Gegner und des Spielers werden dynamisch erzeugt. Falls der Spieler die Schusstaste betätigt, wird die „shoot“-Methode aufgerufen. Hier wird erst überprüft, ob der Spieler genug Munition hat. Falls dies der Fall ist, wird ein neues Objekt vom Typ „PlayerBullet“ erzeugt. Der Konstruktor des Objekts benötigt eine Matrix aus der es die Rotation und Startposition kopieren kann. Ist dies geschehen, wird das Projektil dem Wurzelknoten des Haupt-Szenengraphens hinzugefügt, sodass dieses sich unabhängig künftiger Positions- /Rotationsveränderungen des Spielers bewegen kann. Auf die gleiche Weise funktioniert dies bei der Projektilklasse des Gegners namens „EnemyBullet“.

5. Szenenhierarchie

Alle Objekte erben von der FUDGE-Klasse „Node“, ebenso sind alle Objekte Teil des Wurzelknotens „root“. Die Gegnerklasse hat 4 Kindklassen namens „cacodemonIdle“, „cacodemonHit“, „cacodemonShoot“ und „cacodemonDeath“, welche Teil der Klasse „NodeSprite“ sind.

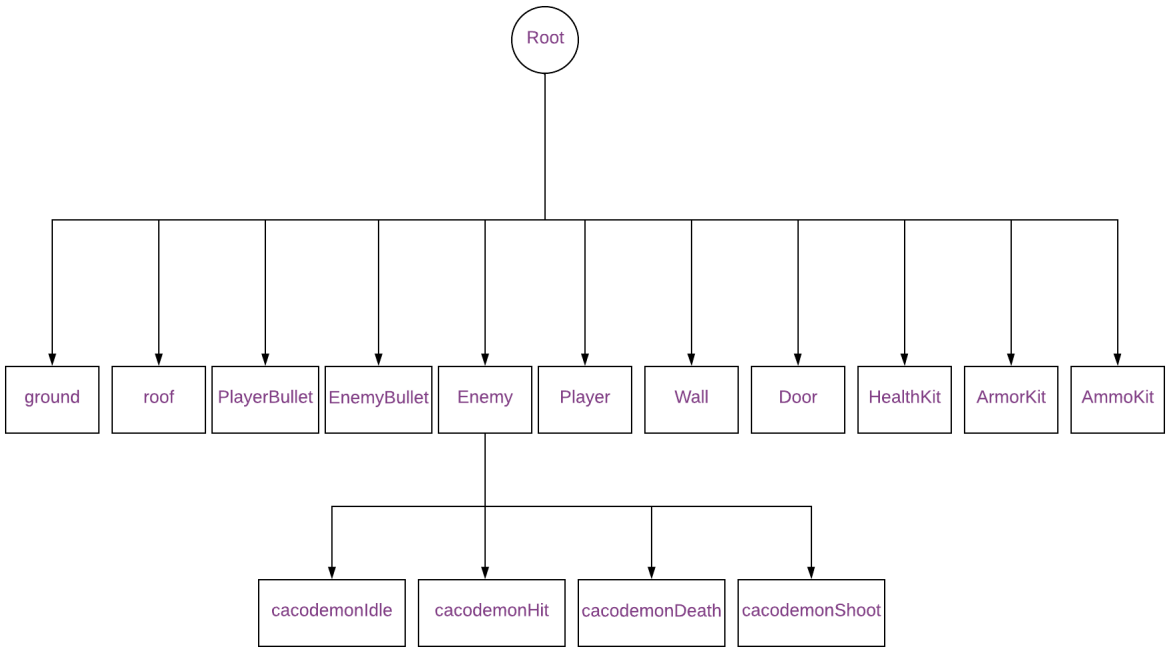

Abbildung 3: Nodebaum

6. Sound

Sound wird mehrfach verwendet im Spiel. So spielt ab Beginn das „Maintheme“ von Doom zur Unterstützung der Atmosphäre. Gegner geben einen Angriffston ab beim Angriff, einen Schadenston um einen Treffer akustisch zu signalisieren und einen Todeston, wenn sie gelöscht werden, um dies dem Spieler zu vermitteln. Türen geben einen Ton beim Öffnen/Schließen, um dem Spieler mehr Feedback zu geben. Spieler geben beim Schießen einen Schusston und bei der Kollision mit einem Gegnerprojektil einen Aufschrei ab. Beim Ableben der Spielfigur wird ein Schmerzensschrei abgeben, wenn der „Todesscreen“ angezeigt wird. Beim Spielsieg wird ein Siegeston abgespielt. Alle genannten Sounds bis auf den Todesschrei, das Maintheme und den Siegeston wurden mit der „ComponentAudio“-Komponente von FUDGE umgesetzt. Demnach bestimmt die Position der Objekte, wie stark ein Sound wahrgenommen wird. Der Spieler hat zum Wahrnehmen der Sounds das „AudioListener“-Objekt.

7. GUI

Zu Beginn des Spiels startet der Spieler im Hauptmenü, hier hat er die Wahl das Spiel zu starten oder die Steuerung unter „Controls“ anzupassen. In diesem Menü kann der Spieler die Bewegungs-, Schuss, Sprint und Interaktionstasten anpassen. Bei einer doppelten Tastenbelegung bekommt der Spieler Feedback und die Taste wird zurückgesetzt. Im Spiel selbst besteht das GUI aus einem HUD mit den Anzeigen Ammo, Health, Stamina, Armor und einem Spielerportrait. Die Anzeigen Ammo und Armor werden textuell dargestellt, während die Anzeigen Health und Stamina durch Balken visualisiert werden. Das Spielerportrait ändert sich je nach Tasteneingabe, so dreht sich das Gesicht nach links, wenn der Spieler sich nach links rotiert. Weiterführend wird eine Interaktionsmeldung auf der Höhe der Pistole angezeigt, wenn sich der Spieler im Interaktionsradius der Tür befindet. In dieser Meldung

wird angezeigt, welche Taste zum Öffnen/Schließen der Tür gedrückt werden kann. Zusätzlich gibt es ein Pausenmenü, ein Todesmenü und ein Gewinnermenü.

Abbildung 4: Ingame HUD

8. Externe Daten

Im LocalStorage des Browsers werden die Tastenbelegungen des „Control“-Menüs gespeichert. Hierdurch kann die Tastenbelegung angepasst werden.

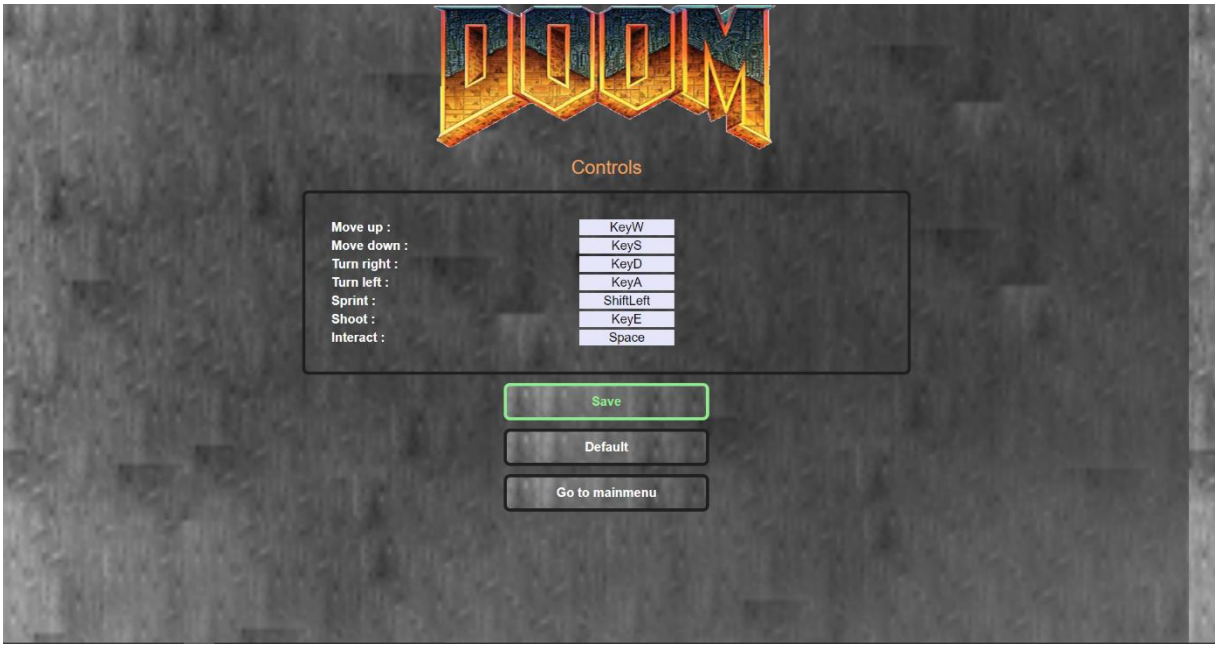

Abbildung 5: Menü zur Tastenbelegung

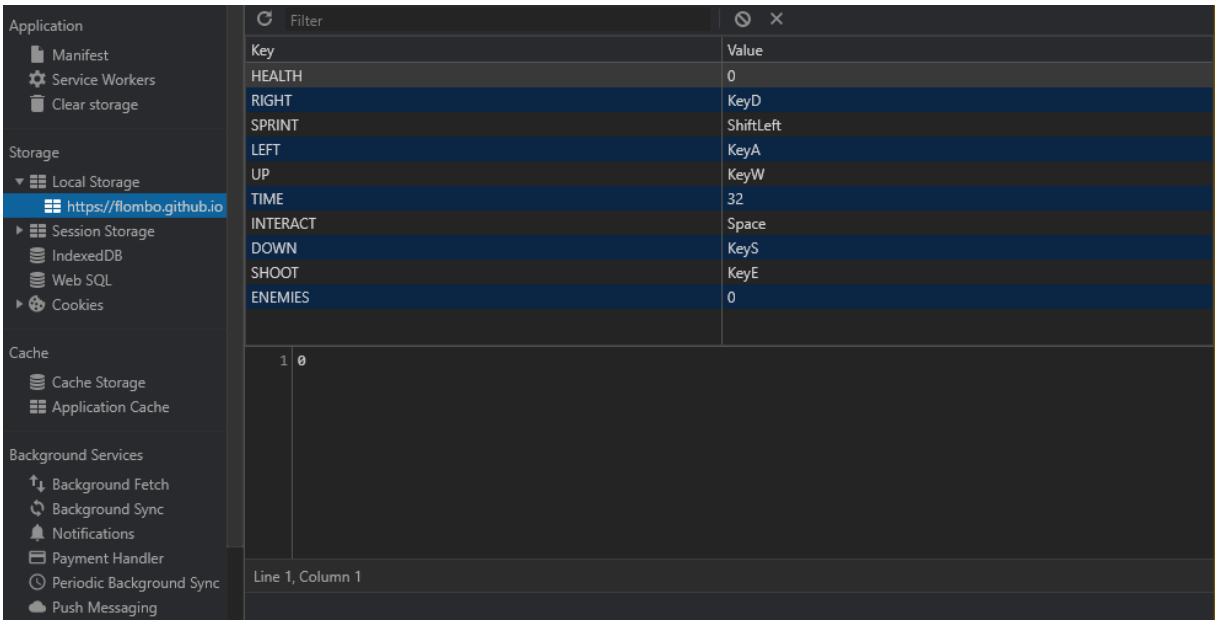

Abbildung 6: Speicherung der Tastenbelegung im LocalStorage

9. Verhaltensklassen

Objekte im Spiel können verschiedene Verhaltensweisen haben. So hat das Gegnerobjekt vier Zustände: „Idle“, „Hunt“, „Attack“, „Flight“. Diese werden durch Radien bestimmt und in der Enumeration-Klasse „EnemyRadius“ gespeichert. Flight hat den kleinsten Radius, Attack einen mittleren und Hunt den größten Radius. Die Spielerposition wird in der Methode „checkPlayerPositionRelativeToRadius“ überprüft, welche als Handlermethode für das „LOOP_FRAME“ –Event dient. Sollte die Spielerposition innerhalb einer der Radien liegen, so wird der dafür zuständige Zustand in der Objektvariabel „currentState“ gesetzt. Ist dies nicht der Fall wird der Zustand Idle gesetzt.

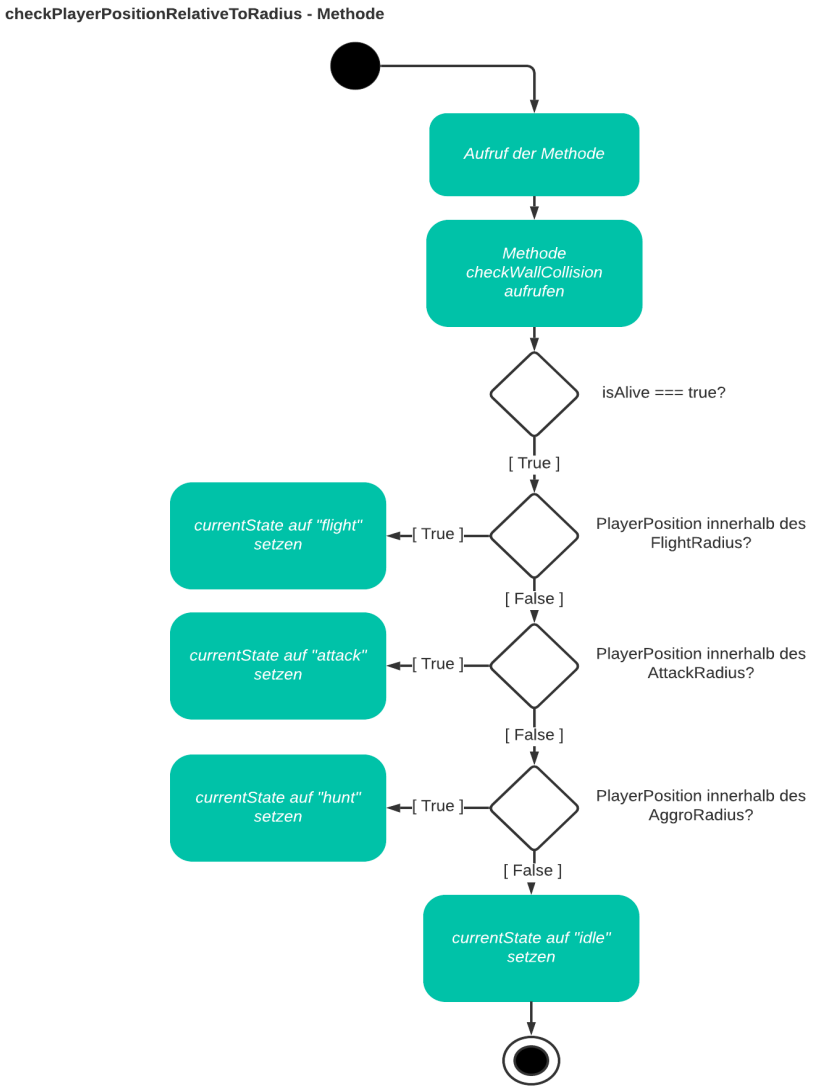

Abbildung 7: checkPlayerPositionRelativeToRadius Aktivitätsdiagramm

Die Zustände werden von der Methode „checkCurrentState“ überprüft, dazu wird am „Loop“-Objekt ein Eventlistener für das Event „LOOP_FRAME“ registriert und die „checkCurrentState“-Methode als Handlermethode definiert. Sollte der Zustand auf Idle gesetzt sein, wird eine Idle-Animation abgespielt. Beim Zustand Hunt bewegt sich der Gegner zum Spieler hin und verfolgt diesen. Liegt der Spieler im Attack-Radius, feuert der Gegner Projektile auf den Spieler und spielt eine Angriffsanimation ab. Sollte der Spieler

innerhalb des Flight-Radius liegen, wechselt der Gegner zu einem Sprite in der Rückansicht und flieht bis er sich im Attack-Radius befindet.

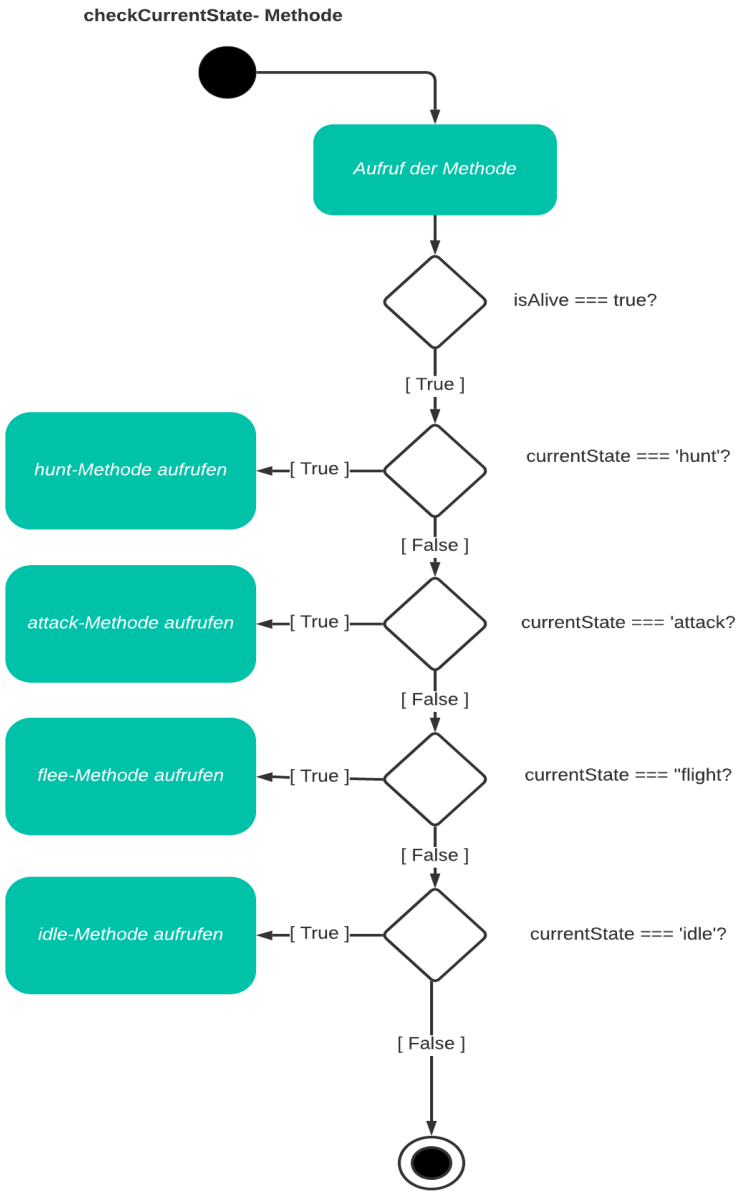

Abbildung 8: checkCurrentState Aktivitätsdiagramm

Eine weitere Verhaltensmethode in der Gegnerklasse wäre die „checkShotCollision“-Methode. Diese ermittelt, ob das Spielerprojektil den Gegner getroffen hat. Ist dies der Fall nimmt der Gegner Schaden, spielt einen Schadenssound ab und eine Schadensanimation.

Das Tür Objekt hat zwei verschiedene Zustände, welche in der Objektvariabel „isClosed“ gespeichert wird. True bedeutet hier, dass die Tür geschlossen ist und False bedeutet eine geöffnete Tür. Das Öffnen und Schließen erfolgt über ein Event, welches vom Spieler gesendet wird, das „playerInteraction“-Event. Dieses Event wird beim Betätigen der Interaktionstaste gefeuert und in der Tür-Klasse durch die Methode „checkPlayerInteraktion“ verarbeitet. Befindet sich der Spieler innerhalb eines bestimmten Radius, so öffnet/schließt sich die Tür.

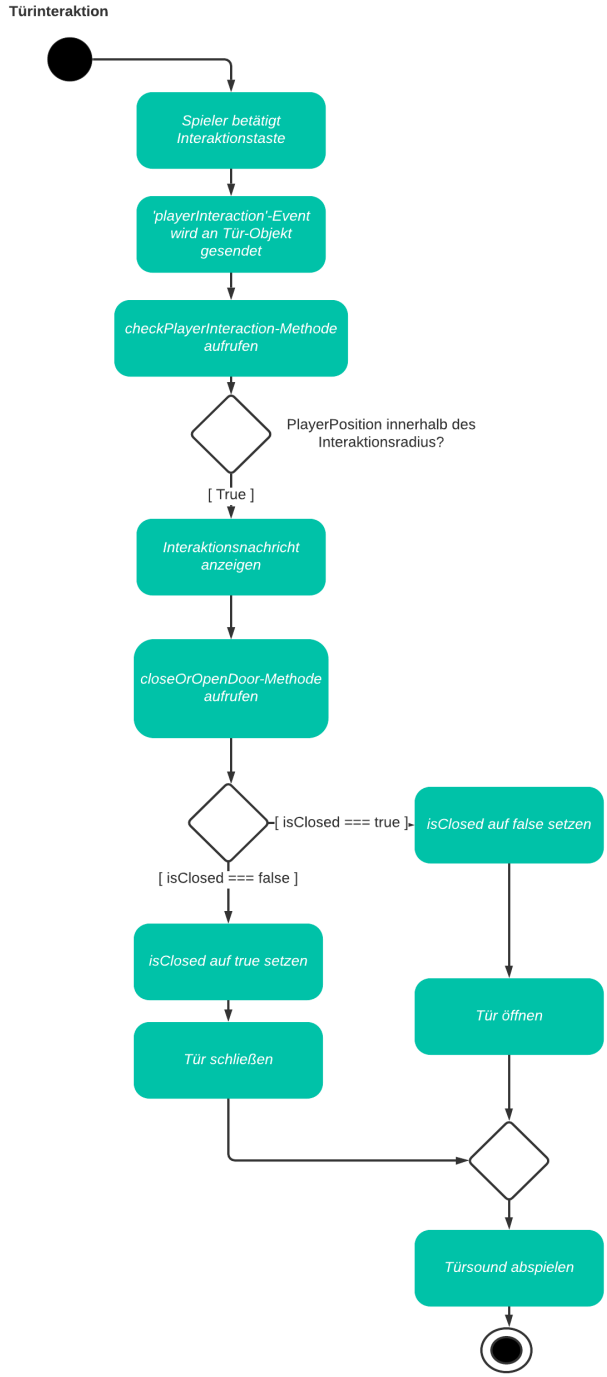

Abbildung 9: Türinteraktion Aktivitätsdiagramm

10.Subklassen

Die Klassen „Wall“ und „Door“ sind Kindklassen der „Obstacle“-Klasse. In der Vaterklasse wird über die Methode „init“ die Position/Rotation, die Komponentenstruktur und die Textur gesetzt. Hierfür benötigt die Methode zwei Paramter „x“ und „z“, welche über den Methodenaufruf im Konstruktor übergeben werden. Der Konstruktor benötigt die Parameter „Player“, „Enemies“, Name, „x“, „z“ und „img“. Die Klasse bietet Getter für das Spielerobjekt und das Gegnerarray. Auch werden in dieser Klasse die Schusskollisionen für die Projektile des Spielers und des Gegners überprüft und verarbeitet. Die beiden Kindklassen „Wall“ und

„Door“ erweitern die Funktionalitäten, indem sie die Spieler- und Gegnerkollisionen überprüfen und verarbeiten. Das „Door“-Objekt hat zudem noch Methoden zum Initialisieren des Sounds für das Öffnen/Schließen, für die Spielerinteraktion (Öffnen/Schließen) und für das Anzeigen einer Interaktionsmeldung, wenn der Spieler nah genug an der Tür steht.

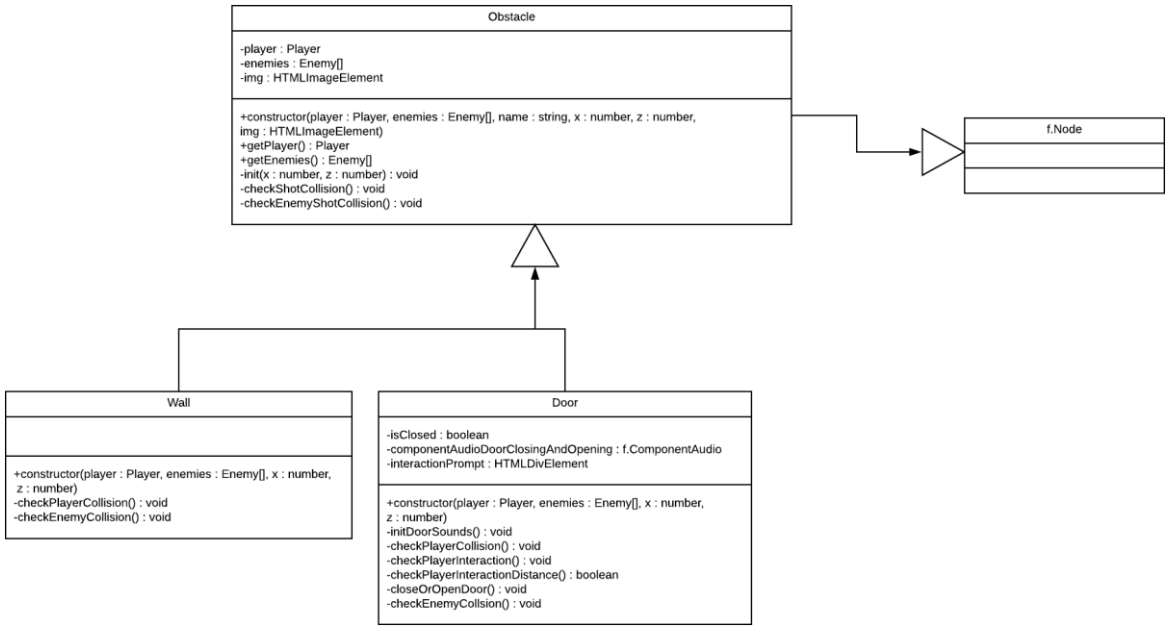

Abbildung 10: Klassendiagramm zu den Klassen Obstacle, Wall und Door

Außerdem sind die Klassen „HealthKit“, „ArmorKit“ und „AmmoKit“ Kindklassen der „Item“-Klasse. In dieser wird in der „init“-Methode die Position, die Komponentenstruktur und die Textur gesetzt. Außerdem wird das Objekt dauerhaft rotiert, durch das registrieren eines Eventlisteners am „Loop“-Objekt von FUDGE. Weitere Methoden sind das registrieren von Kollisionen mit dem Spieler in einer Objektvariable „isColliding“ und eine Funktion zum Löschen des Objekts. Die Kindobjekte erweitern diese Funktionalitäten jeweils durch Hinzufügen von Objektvariablen, welche den Lebenserhalt, den Munitonserhalt und den Rüstungserhalt beim Aufsammeln definieren.

Diese werden jeweils benötigt in der Methode „checkCollision“, in welcher die Kollision mit dem Spieler verarbeitet werden und dem Spieler je nach Item Leben, Rüstung oder Munition hinzugefügt werden. Dazu wird die Objektvariable „isColliding“ des Elternobjekts verwendet.

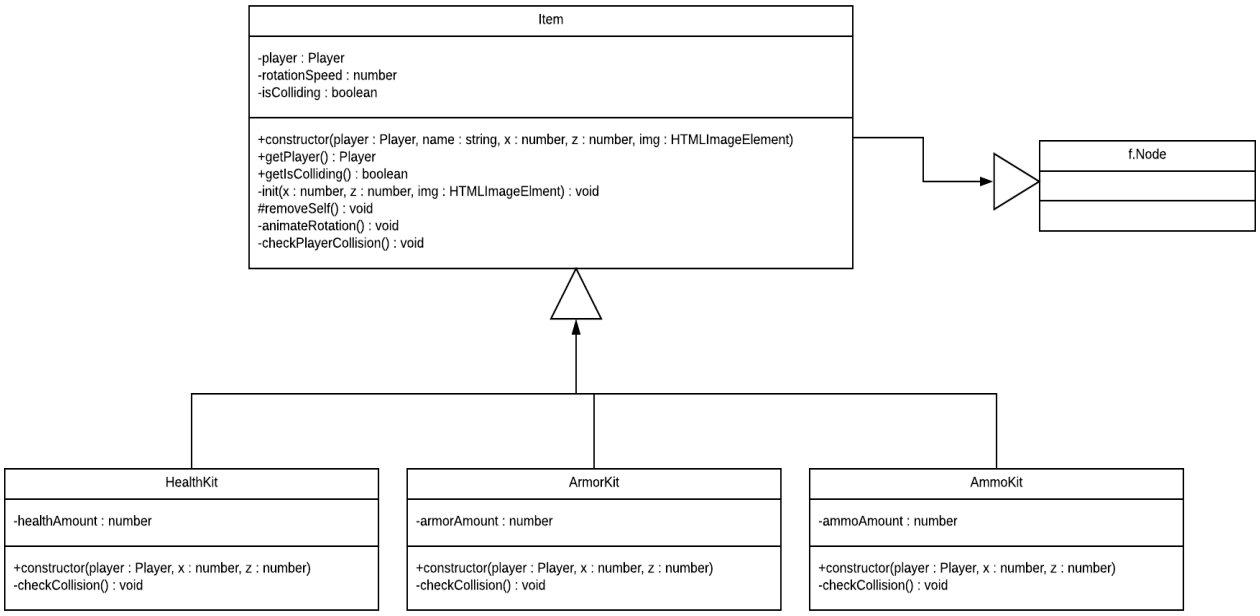

Abbildung 11: Klassendiagramm zu den Klassen Item, HealthKit, ArmorKit und AmmoKit

Weiterführend erben die Klassen „PlayerBullet“ und „EnemyBullet“ vom „Projectiles“-Objekt. Hier ist jedoch die komplette Funktionalität in der Elternklasse vorhanden und nur die verschiedenen Parameter wie Geschwindigkeit, Eventtyp, Startposition, Schadenswert, Range und Verschiebung sind in den Kindobjekten gekapselt.

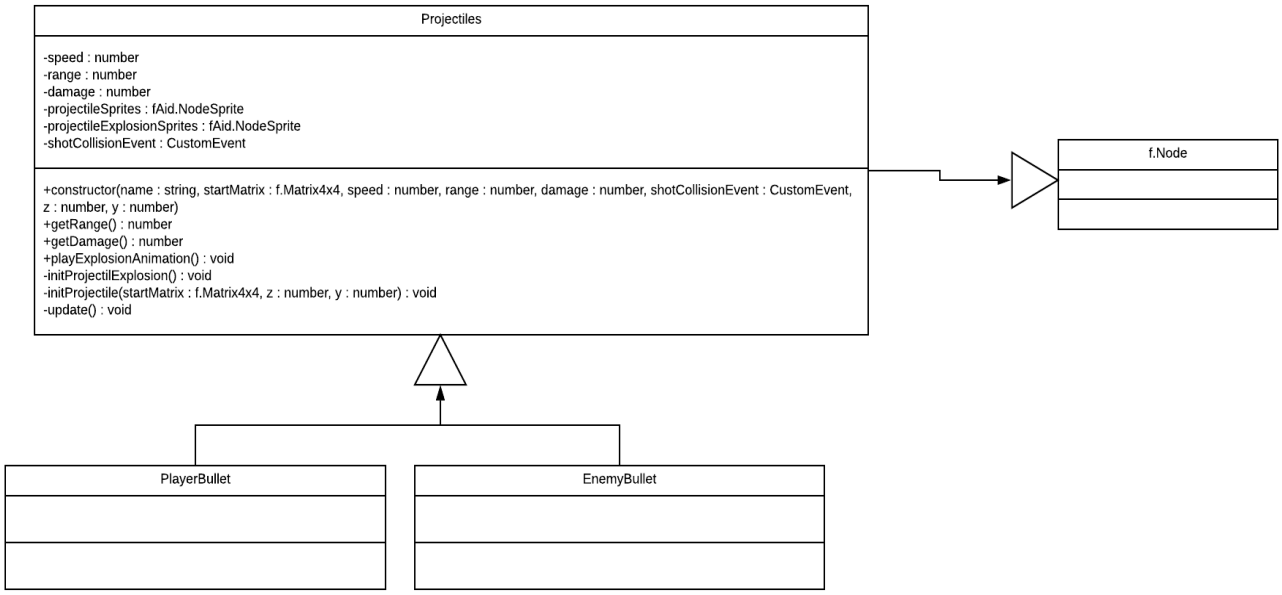

Abbildung 12: Klassendiagramm zu den Klassen Projectiles, EnemyBullet und PlayerBullet

Abschließend ist noch zu sagen, dass alle Objekte im Spiel von der Fudgeklasse „Node“ erben.

11.Maße & Positionen

Alle Spielressourcen befinden sich auf der XZ-Ebene, um möglichst wenig an den Objekten selbst anpassen zu müssen. So konnte einfach die Kamera für die Egoperspektive an die Spielernode gehangen werden, da diese schon vorher auf die XZ-Ebene rotiert wurde. Auch kann so die „lookAt“-Methode fehlerfrei beim Gegnerobjekt aufgerufen werden. Alle beweglichen Objekte wie Spieler, Gegner und Projektile werden nur auf der Z-Achse transliert. Die Ausnahme ist hierbei die Tür, weil diese auf der Y-Achse verschoben werden muss, zum Öffnen/Schließen der Tür. Alle Objekte werden zuerst auf den Ursprung gesetzt, um -90 Grad zuerst auf der Y-Achse und danach auf der Z-Achse rotiert und dann auf der X- und Z-Achse positioniert. Alle Objekte bis auf Wände und die Tür sind nicht skaliert, demnach 1 groß. Die Wände und die Tür sind jeweils um 3 Einheiten auf der Y-Achse und die Wand um 2 Einheiten auf der Z-Achse skaliert. Hierdurch entstehen ein räumliches Empfinden und der Spieler kann problemlos in den geöffneten Raum eintreten.

12.Event-System

Der Spieler sendet über Tastenbefehle Informationen an das Spielerobjekt. Hierzu werden „keyDown“- und „keyUp“-Events verwendet, um in eine Map den keyCode zuschreiben und durch True oder False einen Tastendruck oder ein Loslassen der Taste zu beschreiben. Hierdurch kann das Spielerobjekt sich bewegen, rotieren, schießen, interagieren und sprinten. Bei jeder Bewegung des Spielers, des Gegners und von Projektilen werden CustomEvents versendet. Diese werden beim Empfängerobjekt durch Methoden verarbeitet, beispielsweise für die Kollisionsprüfung oder die Interaktionsprüfung. In dem Tastenbelegungsmenü werden sowohl „mousedown“-Events, als auch „keyDown“-Events verwendet, um Tasten neubelegen zu können.